Claims:

1. (Previously Presented) One or more computer-readable media having stored thereon computer-executable instructions of implementing a kernel emulator for non-native program modules that, when executed by one or more processors, causes the one or more processors to perform a method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel of a native operating system, the non-native kernel calls comprising non-native instructions not operable on the native operating system, the native kernel being part of the native operating system and having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;

converting the intercepted non-native kernel calls into native kernel calls, the native kernel calls comprising native instructions operable on the native operating system and calling the native kernel of the native operating system, the converting comprising translating addresses from non-native length within the non-native kernel calls into native length in the native kernel calls; and

delivering the native kernel calls to the native kernel for direct execution at the native kernel of the operating system without the non-native program modules being re-programmed or re-compiled to operate on the native operating system running the native kernel, thereby facilitating interoperability of the non-native program modules running on the native operating system.

2. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

3. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating non-native CPU instructions into native CPU instructions.

4. (Canceled).

5. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the converting further comprises converting non-native argument format into native argument format.

6. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating words from non-native word size into native word size.

7. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the method further comprises limiting addressable memory to a range addressable by non-native program modules.

8. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the method further comprises managing memory space that is accessible to both native and non-native program modules.

9. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the method further comprises synchronizing a native shared data structure with a non-native shared data structure.

10. (Previously Presented) One or more computer-readable media as recited in claim 1, wherein the method further comprises:

managing memory space accessible to both native and non-native program modules; and

mapping versions of process shared data structures (process SDSs) and versions of thread shared data structures (thread SDSs) between native and the non-native program modules.

11. (Currently amended) One ~~An operating system on the one~~ or more computer-readable media, comprising:

a native kernel configured to receive calls from native program modules; and

a kernel emulator as recited in claim 1 configured to receive and convert calls from non-native program modules for direct handling by the native kernel without the non-native program modules being modified to natively call the native kernel.

12. (Currently amended) One ~~An operating system on the one~~ or more computer-readable media, comprising:

a native kernel configured to receive calls from native APIs;

a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs for direct execution by the native APIs without the non-native APIs being modified to natively utilize the native APIs.

13. (Previously Presented) A method of emulating a kernel for non-native program modules, the method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel of a native operating system, the non-native kernel calls comprising non-native instructions not operable on the native operating system, the native kernel being part of the native operating system and having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;

converting the intercepted non-native kernel calls into native kernel calls, the native kernel calls comprising native instructions operable on the native operating system and calling the native kernel of the native operating system; and

delivering the converted native kernel calls to the native kernel for direct execution at the native kernel of the operating system without the non-native program modules being re-programmed or re-compiled to operate on the native operating system running the native kernel, thereby facilitating interoperability of the non-native program modules running on the native operating system .

14. (Original) A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

15. (Original) A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

16. (Original) A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

17. (Original) A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

18. (Original) A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

19. (Original) A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

20. (Original) A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

21. (Original) A method as recited in claim 20, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

22. (Original) A method as recited in claim 20, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

23. (Original) A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

24. (Original) A method as recited in claim 23, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

25. (Original) A method as recited in claim 23, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

26. (Previously Presented) A computer comprising:

one or more processors; and

memory coupled to the one or more processors, the memory storing thereon computer-executable instructions that, when executed by the one or more processors, implement a kernel emulator for non-native program modules in a native operating system executed on the computer, the kernel emulator being configured to perform a method comprising:

intercepting non-native kernel calls from the non-native program modules, the non-native kernel calls calling a native kernel of a native operating system, the non-native kernel calls comprising non-native instructions not operable on the native operating system, the native kernel being part of the native operating system and having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;

converting the intercepted non-native kernel calls into native kernel calls, the native kernel calls comprising native instructions operable on the native operating system and calling the native kernel of the native operating system; and

delivering the converted native kernel calls to the native kernel for direct execution at the native kernel of the operating system without the non-native program modules being re-programmed or re-compiled to operate on the native operating system.

27-33 (Canceled).

34. (Previously Presented) A method comprising:

emulating a non-native kernel on a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls for direct execution at the native kernel, wherein:

the non-native kernel calls comprises non-native instructions not operable on the native computing platform;

the native kernel is a software component implemented within a native operating system on the native computing platform; and

the emulating facilitates interoperability of the non-native applications calling the native kernel without the non-native applications being re-programmed or re-compiled to operate on the native computing platform on which the non-native applications are not operable to run.

35. (Previously Presented) A method as recited in claim 34, wherein the emulating step comprises:

translating non-native CPU instructions into native CPU instructions;

translating addresses from non-native length into native length; and

limiting addressable memory to a range addressable by non-native program modules.

36. (Original) A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

37. (Previously Presented) A method as recited in claim 34, wherein the converting step comprises translating words from non-native word size into native word size.

38. (Previously Presented) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform a method comprising emulating a non-native kernel on a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls to the native kernel without the non-native applications being modified to target the native computing platform on which the non-native applications are not designed to run.

39. (Canceled).

40. (Currently amended) One or more computer-readable media having stored thereon instructions that ~~implementing a kernel emulator for non-native program modules, the instructions~~, when executed by a computing device having a CPU, configure ~~causing~~ the computing device to perform actions comprising:

implementing a kernel emulator to emulate a non-native kernel on a native computing platform for non-native programs; and

converting ~~so that~~ non-native kernel calls that call a native kernel from non-native applications ~~are converted~~ into native kernel calls to the native kernel to facilitate interoperability of the non-native applications running on the native computing platform without the non-native applications being re-programmed or re-complied for the native computing platform, wherein the non-native kernel calls comprise ~~comprises~~ non-native instructions not operable on the native computing platform.

41. (Currently amended) One or more computer-readable media having stored thereon instructions implementing the kernel emulator recited in claim 40, wherein the ~~instructions of~~ implementing the kernel emulator comprises:

~~instructions~~ implementing an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

~~instructions~~ implementing an address-translator configured to translate addresses from non-native length into native length; and

~~instructions~~ implementing a memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

42. (Currently amended) One or more computer-readable media having stored thereon instructions of an operating system that, when executed on a computing device having a CPU, cause the computing device to ~~implement a plurality of modules, the instructions comprising~~:

implement ~~instructions of implementing~~ a native kernel configured to receive calls from native program modules;

implement ~~instructions of implementing~~ a kernel emulator as recited in claim 40 configured to receive calls from non-native program modules.

43-44. (Canceled).

45. (Currently amended) One or more computer-readable media having stored thereon instructions that, when executed by a computing device having a CPU, ~~causes~~ configure the computing device to perform actions comprising:

~~implement~~ implementing a kernel emulator for non-native program modules, the kernel emulator comprising:

an interceptor configured to intercept non-native kernel calls that call a native kernel from non-native program modules, wherein:

the native kernel is a software component operating system functions and being implemented within a native operating system; and

the non-native kernel calls comprise non-native instructions not operable on the native operating system; and

a call-converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls, the native kernel calls having native instructions calling the native kernel, the call-converter comprising:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length; and

an I/O unit configured to deliver converted native kernel calls to the native kernel,

wherein the call-converter facilitates interoperability of the non-native program modules running on the native platform by enabling the non-native program modules to call the native kernel without the non-native program modules being re-programmed or re-compiled for the native platform running the native kernel on which the non-native program modules are not operable.

46. (Currently amended) ~~An operating system on a computer-readable medium,~~ One or more computer-readable media comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.